Exhibit C

EXHIBIT C

UNITED STATES PATENT NO. 8,948,784
CLAIM CHART FOR INFRINGEMENT OF REPRESENTATIVE INDEPENDENT CLAIM 1

Note: This representative claim chart is provided solely for pleading purposes in this action and is based upon information known at this time. This chart does not represent Plaintiff's infringement contentions, the asserted claims, or all of Plaintiff's allegations regarding infringement. Plaintiff further reserves the right to assert additional or different theories of infringement, including infringement under the doctrine of equivalents.

Infringement analysis provided for any preamble should not be construed as an admission that such preamble is limiting.

US8948784	Google Geofencing API (The accused instrumentality)
1. A computer-implemented method of assessing whether a mobile device is in a geospatial area, the method comprising:	To the extent the preamble is limiting, the accused instrumentality offers a method for assessing (e.g., location of the user) whether a mobile device (e.g., smartphone, laptop, tablet or other handheld device) is in a geospatial area (e.g., through its built-in GPS and other location based sensors). Defendant offers a Geofencing API for mobile app developers to use and enable geo-fence defining capabilities using the user's mobile's built-in sensors such as GPS, accelerometer and other location based sensors. Defining the geo-fences enable the app developers to determine a mobile device context information and provide results or notification the users based on certain interests.

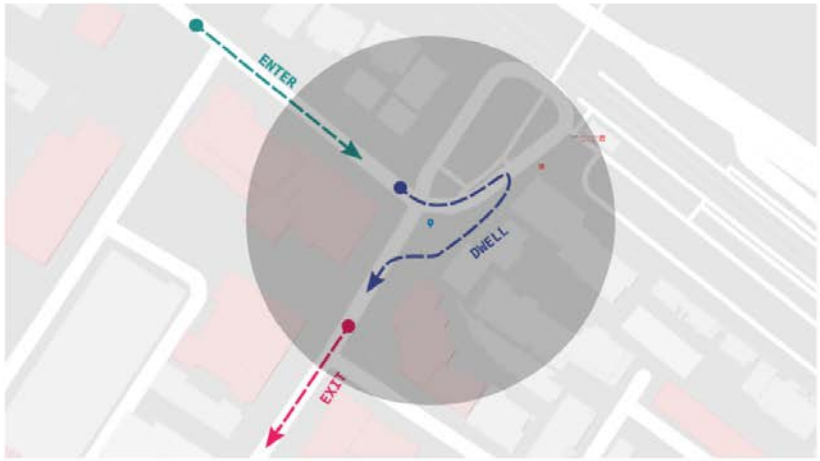
Provide contextual experiences when users enter or leave an area of interest

Sometimes users want to use a particular app when they are in a specific type of location, such as an airport, or a supermarket. However, users have to navigate to that app and then take a specific action within it while they are near the area of interest.

The geofencing API allows you to define perimeters, also referred to as *geofences*, which surround the areas of interest. Your app gets a notification when the device crosses a geofence, which allows you to provide a useful experience when users are in the vicinity.

For example, an airline app can define a geofence around an airport when a flight reservation is near boarding time. When the device crosses the geofence, the app can send a notification that takes users to an activity that allows them to get their boarding pass.

The Geofencing API intelligently uses the device sensors to accurately detect the location of the device in a battery-efficient way.



Receive notifications when users trigger your geofences

Easily create and start monitoring geofences

You can create a list of geofence objects by setting the latitude, longitude, radius, duration, and transition types of each geofence. The transition types indicate the events that trigger the geofence, such as when users enter or exit a geofence.

Once you have a list of geofences, you can add it to a geofencing request. When it's time to start monitoring the geofences, add the request to a geofencing client along with a `PendingIntent` object, which tells the API how to deliver the geofencing events to your app.

Perform an action when users trigger a geofence

The Geofencing API delivers the events to an `IntentService` in your app, which removes the need to have a service running in the background for geofencing purposes. The service is only invoked when there's relevant information.

Your service receives the geofencing event from the `Intent`, including the list of geofences triggered. You can specify your own logic to decide what actions to take.

Source: https://developers.google.com/location-context/geofencing/

Create and monitor geofences

Geofencing combines awareness of the user's current location with awareness of the user's proximity to locations that may be of interest. To mark a location of interest, you specify its latitude and longitude. To adjust the proximity for the location, you add a radius. The latitude, longitude, and radius define a geofence, creating a circular area, or fence, around the location of interest.

You can have multiple active geofences, with a limit of 100 per app, per device user. For each geofence, you can ask Location Services to send you entrance and exit events, or you can specify a duration within the geofence area to wait, or *dwell*, before triggering an event. You can limit the duration of any geofence by specifying an expiration duration in milliseconds. After the geofence expires, Location Services automatically removes it.

Source: https://developer.android.com/training/location/geofencing#CreateAdd

"**Geofencing is a feature in a software program that uses the global positioning system to define geographical boundaries. Combining a user position with a geofence perimeter, it is possible to know if the user is inside or outside the geofence or even if he is exiting or entering the area**.

Geofence API is part of Google's Location APIs. It includes `Geofence`, `GeofencingRequest`, `GeofenceApi`, `GeofencingEvent`, and `GeofenceStatusCodes`.

`Geofence` is an interface that **represents a geographical area that should be monitored**. It is created by using the `Geofence.Builder`. During its creation, you set the monitored region, the geofence's expiration date, responsiveness, an identifier, and the kind of transitions that it should be looking for."

Source: https://en.proft.me/2017/06/6/geographical-boundaries-geofencing-android/ (emphasis added)

<table>
<tr><td></td><td>
<h1>Geofence</h1>
public interface Geofence

Represents a geographical region, also known as a geofence. Geofences can be monitored by geofencer service. And when the user crosses the boundary of a geofence, an alert will be generated.

Nested Class Summary

<table>
<tr><td>class</td><td>Geofence.Builder</td><td>A builder that builds Geofence.</td></tr>
</table>

Source: https://developers.google.com/android/reference/com/google/android/gms/location/Geofence
</td></tr>
<tr><td>receiving, at a mobile device, a representation of a non-circular geospatial zone;</td><td>
The accused instrumentality offers a method to receive a representation of a non-circular geospatial zone (e.g., through Google Maps API and setting a geofence using <code>Geofence.Builder</code> class).

public Geofence.Builder setCircularRegion (double latitude, double longitude, float radius)

Sets the region of this geofence. The geofence represents a circular area on a flat, horizontal plane.

Parameters

<table>
<tr><td>latitude</td><td>latitude in degrees, between -90 and +90 inclusive</td></tr>
<tr><td>longitude</td><td>longitude in degrees, between -180 and +180 inclusive</td></tr>
<tr><td>radius</td><td>radius in meters</td></tr>
</table>

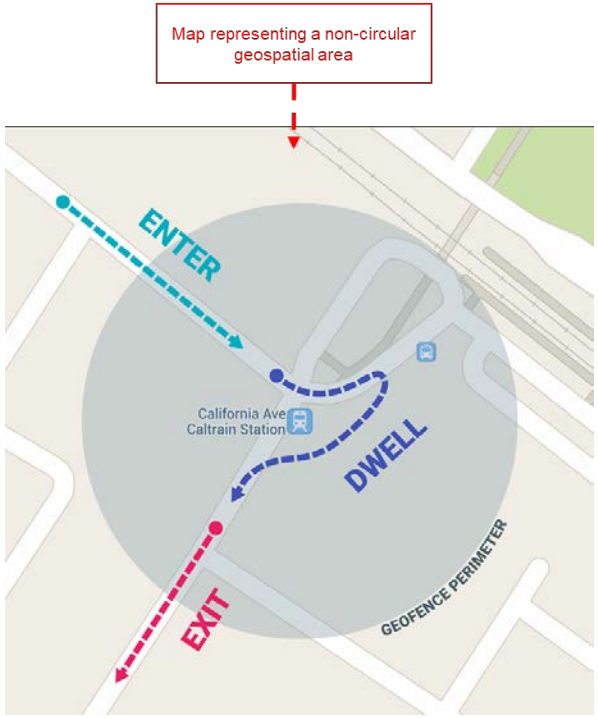
Source:
https://developers.google.com/android/reference/com/google/android/gms/location/Geofence.Builder
</td></tr>
</table>

Add maps

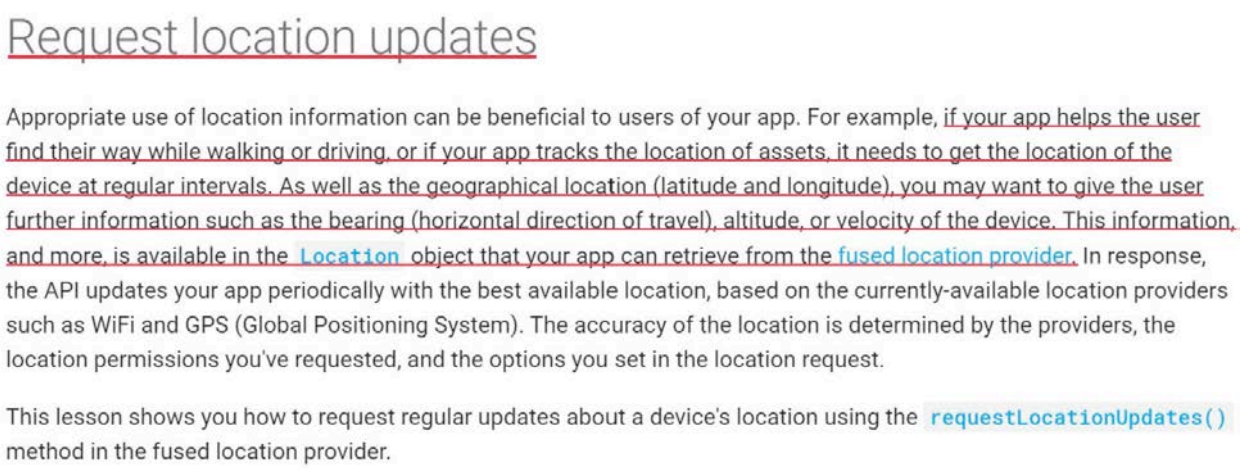
Allow your users to explore the world with rich maps provided by Google. Identify locations with custom markers, augment the map data with image overlays, embed one or more maps as fragments, and much more.

The Google Maps Android API allows you to include maps and customized mapping information in your app.

Source: https://developer.android.com/training/maps/



Source: https://developers.google.com/location-context/geofencing/ (annotated)

	 Source: https://developer.android.com/training/location/request-updates
causing storage of the representation of the non-circular geospatial zone in a memory of the mobile device;	Upon information and belief, the accused instrumentality causes storage of the representation of the non-circular geospatial zone in a memory of the mobile device (e.g., via an instance state stored in a `Bundle` object, Google Maps data layer, app storage using Geofencing API).

Save the state of the activity

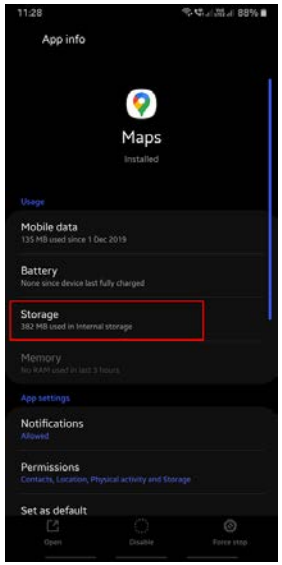
A change to the device's configuration, such as a change in screen orientation or language, can cause the current activity to be destroyed. Your app must therefore store any information it needs to recreate the activity. One way to do this is via an instance state stored in a `Bundle` object.

The following code sample shows how to use the activity's `onSaveInstanceState()` callback to save the instance state:

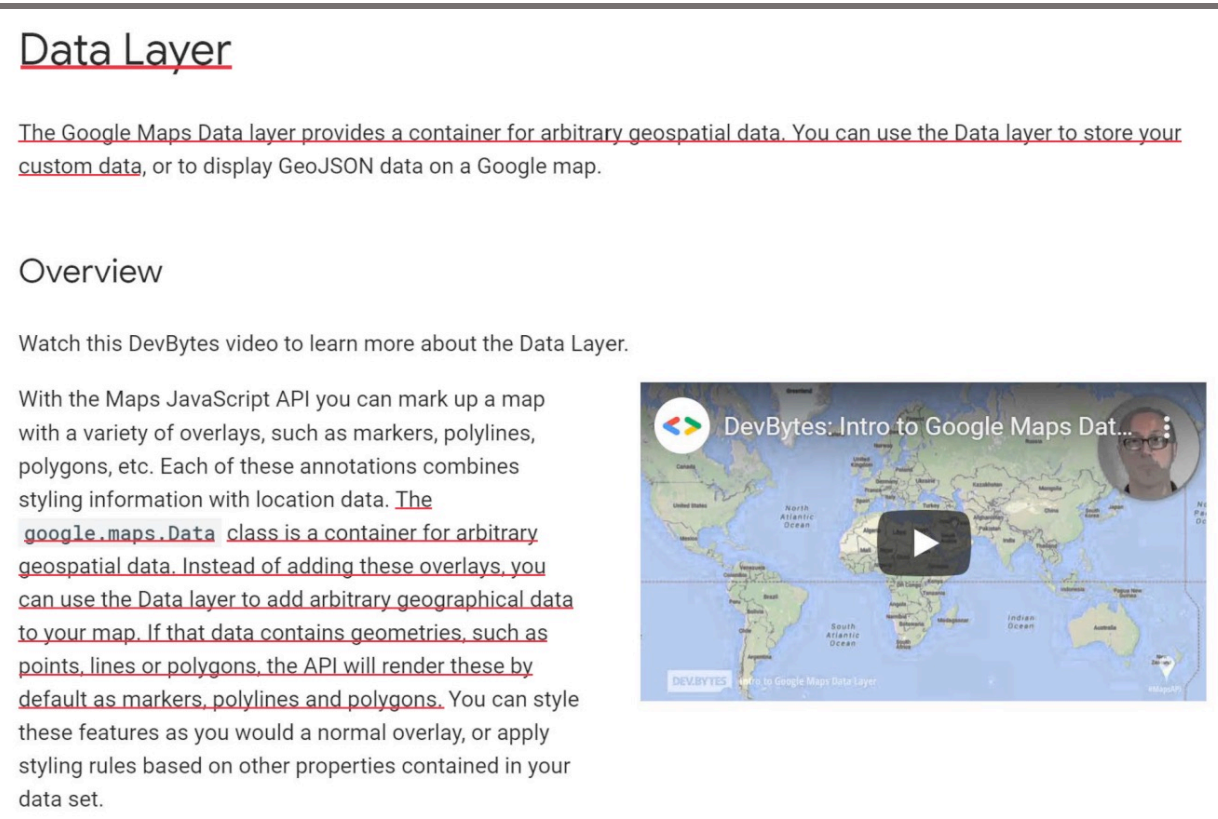
KOTLIN JAVA

```
override fun onSaveInstanceState(outState: Bundle?) {
    outState?.putBoolean(REQUESTING_LOCATION_UPDATES_KEY, requestingLocationUpdates)
    super.onSaveInstanceState(outState)
}
```

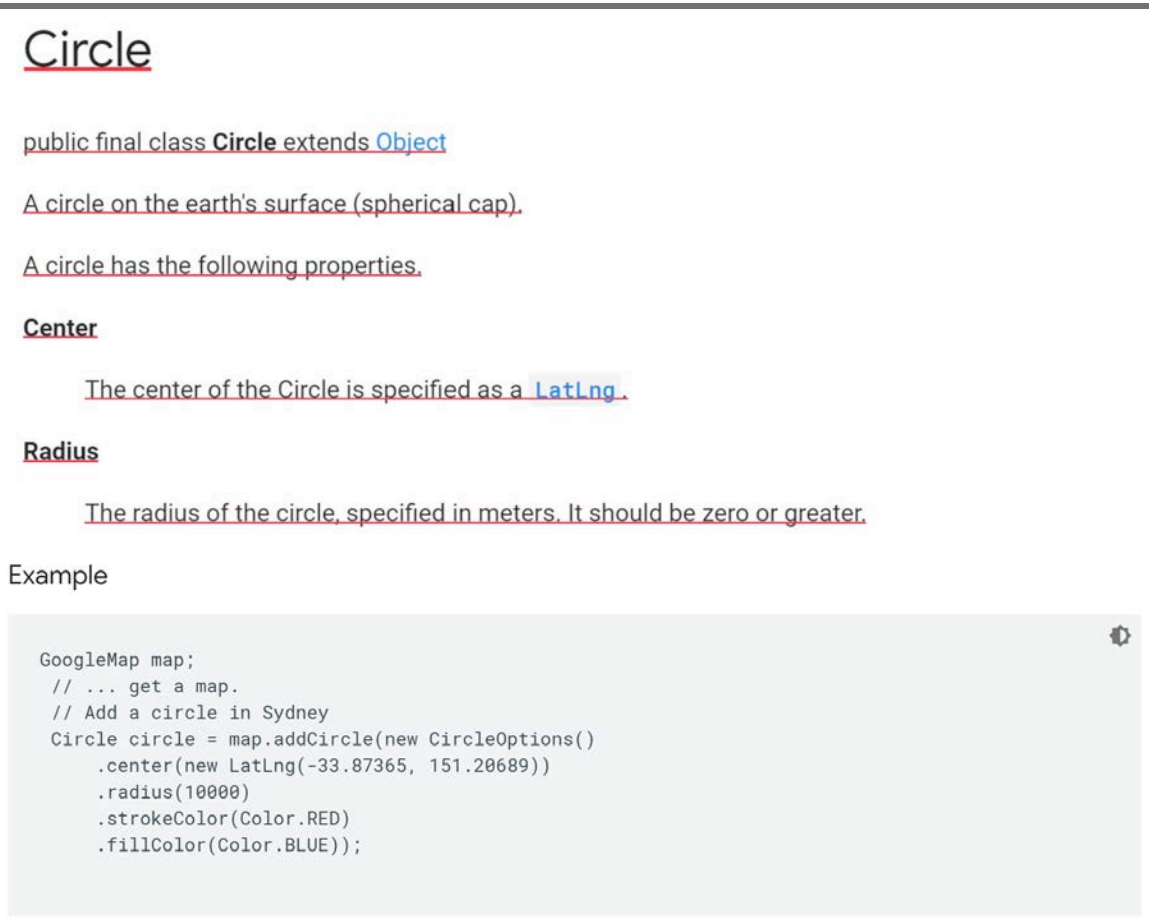
Source: https://developer.android.com/training/location/request-updates#save-state



Source: Actual Screenshot (Android 10)

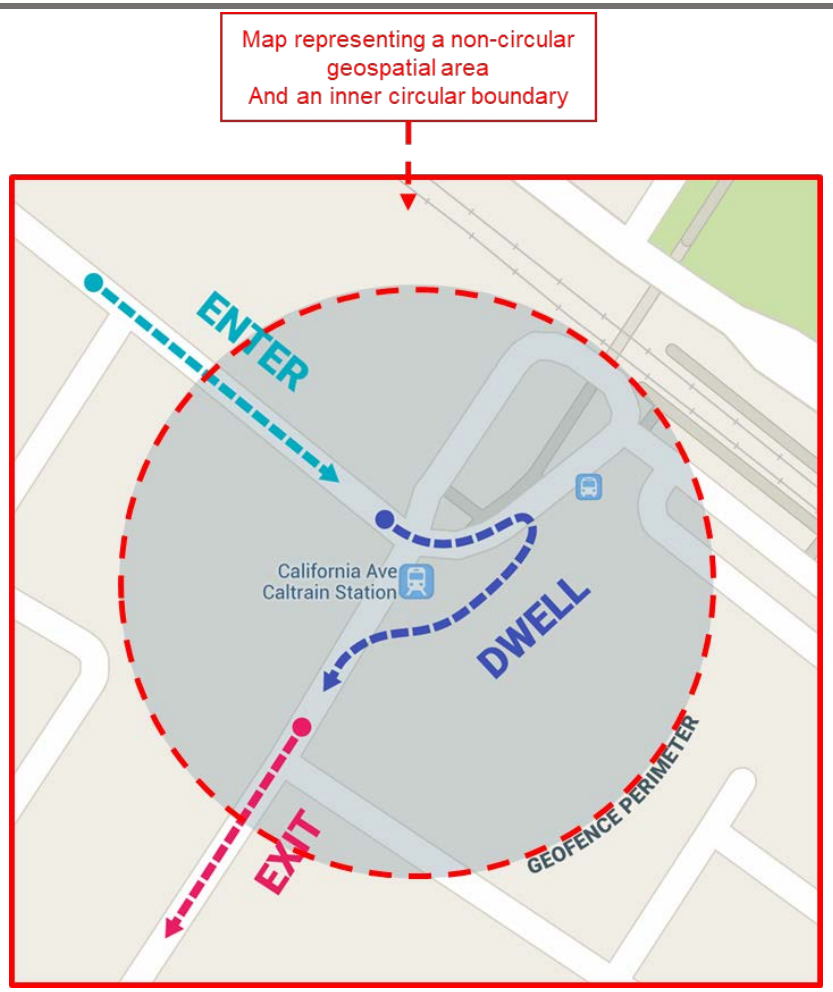
	 Source: https://developers.google.com/maps/documentation/javascript/datalayer
determining, based at least in part on the non-circular geospatial zone, an outer circular boundary entirely encompassing the non-circular geospatial zone, wherein the outer circular boundary is	The accused instrumentality determines based at least in part on the non-circular geospatial zone (e.g., Maps) an outer circular boundary entirely encompassing the non-circular geospatial zone wherein the outer circular boundary is defined by at least a centroid of the non-circular geospatial zone and a maximum radius (e.g., Circle class part of Location reference APIs from google. This class is defined as being a circle on the earth's surface (spherical cap)).

defined by at least a centroid of the non-circular geospatial zone and a maximum radius;	**public** Geofence.Builder **setCircularRegion** (double latitude, double longitude, float radius) Sets the region of this geofence. The geofence represents a circular area on a flat, horizontal plane. **Parameters** **latitude** — latitude in degrees, between -90 and +90 inclusive **longitude** — longitude in degrees, between -180 and +180 inclusive **radius** — radius in meters Source: https://developers.google.com/android/reference/com/google/android/gms/location/Geofence.Builder Add maps Allow your users to explore the world with rich maps provided by Google. Identify locations with custom markers, augment the map data with image overlays, embed one or more maps as fragments, and much more. The Google Maps Android API allows you to include maps and customized mapping information in your app.  Source: https://developer.android.com/training/maps/

	 Source: https://developers.google.com/android/reference/com/google/android/gms/maps/model/Circle
determining, based at least in part on the non-circular geospatial zone, an inner circular boundary entirely encompassed within	The accused instrumentality determines based at least in part on the non-circular geospatial zone (e.g., Maps) an inner circular boundary entirely encompassed within the non-circular geospatial zone, wherein the inner circular boundary is defined by at least the centroid of the non-circular geospatial zone and a minimum radius (e.g., boundary of geofence is set by adjusting the latitude, longitude, centroid and minimum radius.)

the non-circular geospatial zone, wherein the inner circular boundary is defined by at least the centroid of the non-circular geospatial zone and a minimum radius;	First, use `Geofence.Builder` to create a geofence, setting the desired radius, duration, and transition types for the geofence. For example, to populate a list object: KOTLIN JAVA ``` geofenceList.add(new Geofence.Builder() // Set the request ID of the geofence. This is a string to identify this // geofence. .setRequestId(entry.getKey()) .setCircularRegion(entry.getValue().latitude, entry.getValue().longitude, Constants.GEOFENCE_RADIUS_IN_METERS) .setExpirationDuration(Constants.GEOFENCE_EXPIRATION_IN_MILLISECONDS) .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER | Geofence.GEOFENCE_TRANSITION_EXIT) .build()); ``` This example pulls data from a constants file. In actual practice, apps might dynamically create geofences based on the user's location. Source: https://developer.android.com/training/location/geofencing#java Easily create and start monitoring geofences You can create a list of geofence objects by setting the latitude, longitude, radius, duration, and transition types of each geofence. The transition types indicate the events that trigger the geofence, such as when users enter or exit a geofence. Once you have a list of geofences, you can add it to a geofencing request. When it's time to start monitoring the geofences, add the request to a geofencing client along with a `PendingIntent` object, which tells the API how to deliver the geofencing events to your app. Source: https://developers.google.com/location-context/geofencing

	## Create and monitor geofences Geofencing combines awareness of the user's current location with awareness of the user's proximity to locations that may be of interest. To mark a location of interest, you specify its latitude and longitude. To adjust the proximity for the location, you add a radius. The latitude, longitude, and radius define a geofence, creating a circular area, or fence, around the location of interest. You can have multiple active geofences, with a limit of 100 per app, per device user. For each geofence, you can ask Location Services to send you entrance and exit events, or you can specify a duration within the geofence area to wait, or *dwell*, before triggering an event. You can limit the duration of any geofence by specifying an expiration duration in milliseconds. After the geofence expires, Location Services automatically removes it. Source: https://developer.android.com/training/location/geofencing#CreateAdd



Source: https://developer.android.com/training/location/geofencing

	The geofencing API allows you to define perimeters, also referred to as *geofences*, which surround the areas of interest. Your app gets a notification when the device crosses a geofence, which allows you to provide a useful experience when users are in the vicinity. Source: https://developers.google.com/location-context/geofencing/ Choose the optimal radius for your geofence For best results, the minimium radius of the geofence should be set between 100 - 150 meters. When Wi-Fi is available location accuracy is usually between 20 - 50 meters. When indoor location is available, the accuracy range can be as small as 5 meters. Unless you know indoor location is available inside the geofence, assume that Wi-Fi location accuracy is about 50 meters. When Wi-Fi location isn't available (for example, when you are driving in rural areas) the location accuracy degrades. The accuracy range can be as large as several hundred meters to several kilometers. In cases like this, you should create geofences using a larger radius. Source: https://developer.android.com/training/location/geofencing#choose-the-optimal-radius-for-your-geofenceAnalyst
computing a current distance of the mobile device from the centroid of the non-circular geospatial zone; and	The accused instrumentality computes current distance of the mobile device from the centroid of the non-circular geospatial zone (e.g., the current distance of mobile device from the centroid of the non-circular geospatial zone is measured using latitude, longitude and radius parameters of the geofence builder class). Set up for geofence monitoring The first step in requesting geofence monitoring is to request the necessary permission. To use geofencing, your app must request `ACCESS_FINE_LOCATION`. If your app targets Android 10 (API level 29) or higher, your app must also request `ACCESS_BACKGROUND_LOCATION`. Source: https://developer.android.com/training/location/geofencing#RequestGeofences

Create and monitor geofences

Geofencing combines awareness of the user's current location with awareness of the user's proximity to locations that may be of interest. To mark a location of interest, you specify its latitude and longitude. To adjust the proximity for the location, you add a radius. The latitude, longitude, and radius define a geofence, creating a circular area, or fence, around the location of interest.

You can have multiple active geofences, with a limit of 100 per app, per device user. For each geofence, you can ask Location Services to send you entrance and exit events, or you can specify a duration within the geofence area to wait, or *dwell*, before triggering an event. You can limit the duration of any geofence by specifying an expiration duration in milliseconds. After the geofence expires, Location Services automatically removes it.

Source: https://developer.android.com/training/location/geofencing#CreateAdd

The Geofencing API intelligently uses the device sensors to accurately detect the location of the device in a battery-efficient way.

Source: https://developers.google.com/location-context/geofencing/

The geofencing API allows you to define perimeters, also referred to as *geofences*, which surround the areas of interest. Your app gets a notification when the device crosses a geofence, which allows you to provide a useful experience when users are in the vicinity.

Source: https://developers.google.com/location-context/geofencing/

Choose the optimal radius for your geofence

For best results, the minimum radius of the geofence should be set between 100 - 150 meters. When Wi-Fi is available location accuracy is usually between 20 - 50 meters. When indoor location is available, the accuracy range can be as small as 5 meters. Unless you know indoor location is available inside the geofence, assume that Wi-Fi location accuracy is about 50 meters.

When Wi-Fi location isn't available (for example, when you are driving in rural areas) the location accuracy degrades. The accuracy range can be as large as several hundred meters to several kilometers. In cases like this, you should create geofences using a larger radius.

Source: https://developer.android.com/training/location/geofencing#choose-the-optimal-radius-for-your-geofenceAnalyst

Geofence

public interface **Geofence**

Represents a geographical region, also known as a geofence. Geofences can be monitored by geofencer service. And when the user crosses the boundary of a geofence, an alert will be generated.

Nested Class Summary

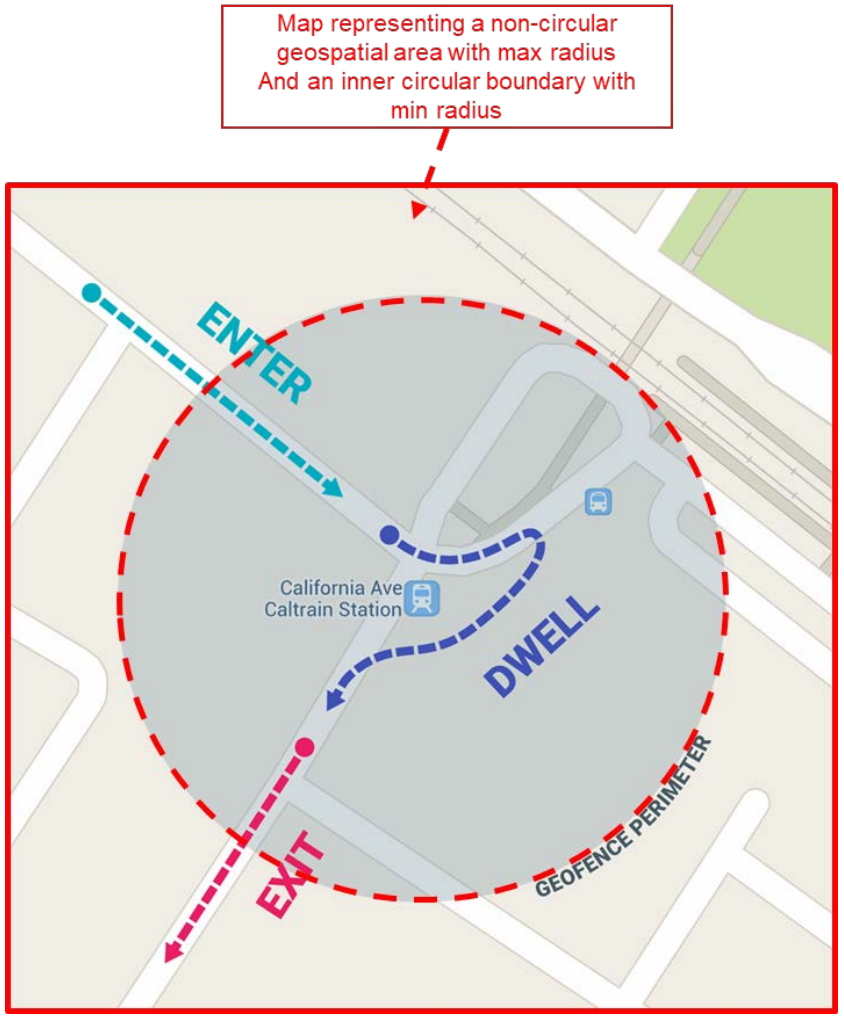
class	Geofence.Builder	A builder that builds `Geofence`.

Constant Summary

int	GEOFENCE_TRANSITION_DWELL	The transition type indicating that the user enters and dwells in geofences for a given period of time.
int	GEOFENCE_TRANSITION_ENTER	The transition type indicating that the user enters the geofence(s).
int	GEOFENCE_TRANSITION_EXIT	The transition type indicating that the user exits the geofence(s).
long	NEVER_EXPIRE	Expiration value that indicates the geofence should never expire.

Source: https://developers.google.com/android/reference/com/google/android/gms/location/Geofence

in response to a determination that the computed current distance both exceeds the minimum radius and does not exceed the maximum radius, utilizing the non-circular geospatial zone to determine a context of the mobile device.	The accused instrumentality determines that the computed current distance both exceeds the minimum radius and does not exceed the maximum radius and utilizes the non-circular geospatial zone to determine a context of the mobile device (e.g., based on location and motion (context) of the mobile device, if the users location triggers geofence boundary, notifications may be provided to the users on their device.) Create and monitor geofences Geofencing combines awareness of the user's current location with awareness of the user's proximity to locations that may be of interest. To mark a location of interest, you specify its latitude and longitude. To adjust the proximity for the location, you add a radius. The latitude, longitude, and radius define a geofence, creating a circular area, or fence, around the location of interest. You can have multiple active geofences, with a limit of 100 per app, per device user. For each geofence, you can ask Location Services to send you entrance and exit events, or you can specify a duration within the geofence area to wait, or *dwell*, before triggering an event. You can limit the duration of any geofence by specifying an expiration duration in milliseconds. After the geofence expires, Location Services automatically removes it.



Source: https://developer.android.com/training/location/geofencing#CreateAdd (annotated)

Geofence

public interface **Geofence**

Represents a geographical region, also known as a geofence. Geofences can be monitored by geofencer service. And when the user crosses the boundary of a geofence, an alert will be generated.

Nested Class Summary

class	Geofence.Builder	A builder that builds `Geofence`.

Constant Summary

int	GEOFENCE_TRANSITION_DWELL	The transition type indicating that the user enters and dwells in geofences for a given period of time.
int	GEOFENCE_TRANSITION_ENTER	The transition type indicating that the user enters the geofence(s).
int	GEOFENCE_TRANSITION_EXIT	The transition type indicating that the user exits the geofence(s).
long	NEVER_EXPIRE	Expiration value that indicates the geofence should never expire.

Source: https://developers.google.com/android/reference/com/google/android/gms/location/Geofence

Specify geofences and initial triggers

The following snippet uses the `GeofencingRequest` class and its nested `GeofencingRequestBuilder` class to specify the geofences to monitor and to set how related geofence events are triggered:

KOTLIN JAVA

```
private GeofencingRequest getGeofencingRequest() {
    GeofencingRequest.Builder builder = new GeofencingRequest.Builder();
    builder.setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER);
    builder.addGeofences(geofenceList);
    return builder.build();
}
```

This example shows the use of two geofence triggers. The `GEOFENCE_TRANSITION_ENTER` transition triggers when a device enters a geofence, and the `GEOFENCE_TRANSITION_EXIT` transition triggers when a device exits a geofence. Specifying `INITIAL_TRIGGER_ENTER` tells Location services that `GEOFENCE_TRANSITION_ENTER` should be triggered if the device is already inside the geofence.

In many cases, it may be preferable to use instead `INITIAL_TRIGGER_DWELL`, which triggers events only when the user stops for a defined duration within a geofence. This approach can help reduce "alert spam" resulting from large numbers notifications when a device briefly enters and exits geofences. Another strategy for getting best results from your geofences is to set a minimum radius of 100 meters. This helps account for the location accuracy of typical Wi-Fi networks, and also helps reduce device power consumption.

Source: https://developer.android.com/training/location/geofencing#java